Comparing basic collection functionalities between Swift, Kotlin and Java

Collection type	Swift	Kotlin	Java 8
LIST / ARRAY			
Static Array	**[…]** or **Array**<…>	…Array or **Array**<…>	**…[]**
Int - example	[Int] or Array<Int>	Example	int[]
	var array: [Int] = [1, 2, 3]	var array: IntArray = intArrayOf(1, 2, 3) or var array: Array<Int> = arrayOf(1, 2, 3)	int[] array = new int[] {1, 2, 3};
	array[0] = 1	array[0] = 1	array[0] = 1;
	let x = array[0]	val x = array[0]	int x = array[0];
	let l = array.count	val l = array.count()	int l = array.length
	for i in array { print(i) }	for (i in array) { println(i) }	for (int i : array) { System.out.println(i); }
Dynamic Array Fixed Size	**[…]** or **Array**<…>	…Array or **Array**<…>	**…[]**
Int - example	[Int] or Array<Int>	…Array	int[]
	var array: [Int] = Array<Int>(repeating: 0, count: 3)	val array:IntArray = IntArray(3) or var array: Array<Int> = Array<Int>(3) { 0 }	int[] array = new int[3];
	array[0] = 1	array[0] = 1	array[0] = 1;
	let x = array[0]	val x = array[0]	int x = array[0];
	let l = array.count	val l = array.count()	int l = array.length
	for i in array { print(i) }	for (i in array) { println(i) }	for (int i : array) { System.out.println(i) }
Dynamic Array Dynamic Size	[…] or **Array**<…>	**MutableList**<…>	**ArrayList**<…>
Int - mutable example	[Int] or Array<Int>	MutableList<Int>	ArrayList<Integer>
	var array = [Int]()	val array = mutableListOf<Int>()	ArrayList<Integer> array = new ArrayList();
	var array: [Int] = [1, 2, 3]	val array: MutableList<Int> = mutableListOf(1, 2, 3)	ArrayList<Integer> array = new ArrayList<Integer>(Arrays.asList(1, 2, 3));
	var array: [Int] = Array<Int>(repeating: 0, count: 3)	val array: MutableList<Int> = MutableList(3) { 0 }	ArrayList<Integer> array = new ArrayList<Integer>(Collections.nCopies(3, 0));
	array.append(4)	array.add(4)	array.add(4);
	array.insert(5, at: 0)	array.add(0,5)	array.add(0,5);
SET			
Dynamic Set	**Set**<…>	**MutableSet**<…>	**HashSet**<…>
Int - mutable example	Set<Int>	MutableSet<Int>	HashSet<Integer>
	var set = Set<Int>()	val set:MutableSet<Int> = hashSetOf<Int>()	Set<Integer> set = new HashSet<Integer>();
	var set: Set<Int> = [1, 2, 3]	val set = hashSetOf(1, 2, 3)	Set<Integer> set = new HashSet<Integer>(Arrays.asList(1, 2, 3));
	set.insert(4)	set.add(4)	set.add(4);
	set.remove(3)	set.remove(3)	set.remove(3);
	if set.contains(3) { … }	if (set.contains(3)) { … }	if (set.contains(3)) { … }
MAP / DICTIONARY			
Dynamic Map	**Dictionary**<…, ..>	**MutableMap**<…, …>	**HashMap**<…, …>
Int, String - mutable example	Dictionary<Int, String>	MutableMap<Int, String>	HashMap<Integer, String>
	var map = Dictionary<Int, String>()	val map:MutableMap<Int, String> = mutableMapOf<Int, String>()	Map<Integer, String> map = new HashMap<Integer, String>();
	var map = [1: "one", 2: "two", 3: "three"]	val map = mutableMapOf(1 to "one", 2 to "two", 3 to "three")	//only Guava or Java 9 with Map<Int, String> map = Map.ofEntries(entry(1, “one”), …);
	map[4] = "four"	map[4] = "four"	map.put(4, “four”); // map.get(4);
	for (key, value) in map { print("\(key)-\(value)") }	for((key, value) in map) { println("$key-$value") }	for(Map.Entry e : map.entrySet()) { System.out.println(e.getKey()+"-"+e.getValue()); }